Portal Integration

The Portlet API defines the way a portal server assembles pages out of content fragments that are provided by different applications. The Portlet API was published in its 1.0 version in October 2003 as JSR (Java Specification Request) 168. See the Java Community Process pages at *http://jcp.org/* for details. It is very important to point out the Portlet 1.0 Errata document which was published in May 2005 and which is an addition to the 1.0 standard. See *http://jcp.org/aboutJava/communityprocess/maintenance/jsr168/Portlet1.0-ERRATA.html* for details.

Application Designer provides an integration into portal servers that is fully compliant with the Portlet 1.0 standard. It allows you to easily import any page that is built using Application Designer into portal scenarios, i.e. the page can directly be used as part of a portal page.

The portal integration features are:

- Any page can be wrapped into a portal without coding effort.
- Parameters can be passed into the page by the normal URL extension (“&name=value”), by a POST or GET request.
- The application has full read and write access to the portlet request (and by this to the portlet session and to the portlet context) and thus can share and exchange data with other portlets.
- The deployment units (*.war* files) built by Application Designer are directly usable as input by portlet servers.

The use of the Portlet API is described in the following sections:

- Integrating Pages as Portlets
- Session Management and Portlet API Support
- Portlet Integration and AJAX

Integrating Pages as Portlets

There is one generic portlet that comes with Application Designer. The name is:

```
com.softwareag.cis.server.PortletWrapper
```

The portlet allows you to pass portlet preferences:

- PAGEURL - this is the name of the Application Designer page that is opened inside the portlet. The format is "/project/page.html".
- NORMWIDTH/NORMHEIGHT - the width and height (as px/pt/% value) that the portlet should have when opened in *normal size*. The default is 400 for width and 300 for height.
- MAXWIDTH/MAXHEIGHT - the width and height (as px/pt/% value) that the portlet should have when opened in *maximum size*. The default is 100% both for width and height.

The specification of the portlet preferences needs to be done following the description of your portlet server.

Example: when deploying an Application Designer to the portlet reference implementation (Apache Pluto), then the portlets are registered in the *portletentityregistry.xml* file in the following way:

```
<application id="8">
    <definition-id>cis</definition-id>
    <portlet id="1">
        <definition-id>cis.PortletWrapper</definition-id>
        <preferences>
            <pref-name>PAGEURL</pref-name>
            <pref-value>/HTMLBasedGUI/com.softwareag.cis.admin.serverlog.html</pref-value>
            <read-only>true</read-only>
        </preferences>
        <preferences>
            <pref-name>MAXHEIGHT</pref-name>
            <pref-value>600</pref-value>
            <read-only>true</read-only>
        </preferences>
        <preferences>
            <pref-name>MAXWIDTH</pref-name>
            <pref-value>100%</pref-value>
            <read-only>true</read-only>
        </preferences>
    </portlet>
    <portlet id="2">
        <definition-id>cis.PortletWrapper</definition-id>
        <preferences>
            <pref-name>PAGEURL</pref-name>
            <pref-value>/HTMLBasedGUI/com.softwareag.cis.workplace.logon.html</pref-value>
            <read-only>true</read-only>
        </preferences>
        <preferences>
            <pref-name>MAXHEIGHT</pref-name>
            <pref-value>600</pref-value>
            <read-only>true</read-only>
        </preferences>
        <preferences>
            <pref-name>MAXWIDTH</pref-name>
            <pref-value>100%</pref-value>
            <read-only>true</read-only>
        </preferences>
    </portlet>
    <portlet id="3">
        <definition-id>cis.PortletWrapper</definition-id>
        <preferences>
            <pref-name>PAGEURL</pref-name>
            <pref-value>/HTMLBasedGUI/xyz.html</pref-value>
            <read-only>true</read-only>
        </preferences>
        <preferences>
            <pref-name>MAXHEIGHT</pref-name>
            <pref-value>600</pref-value>
            <read-only>true</read-only>
        </preferences>
        <preferences>
            <pref-name>MAXWIDTH</pref-name>
            <pref-value>100%</pref-value>
            <read-only>true</read-only>
        </preferences>
    </portlet>
</application>
```

The web application's name is *cis*. You see that per portlet a portlet definition is made, each definition pointing to the PAGEURL that it should open and each definition specifying the MAXHEIGHT and MAXWIDTH. In the Pluto environment the defined portlets can now be addressed from the portal page definition file (*pageregistry.xml*).

Session Management and Portlet API Support

The portal server is responsible for a certain session management (and e.g. for single signon support within this session management).

Application Designer comes with its own session management. The coupling is done in the following way: the portal's session management is seen as the "Master Session Management". This means that sessions are opened inside Application Designer at the point of time when they are first accessed through a portlet. The session ID that is used inside Application Designer is the same as the session ID that comes from the portal server. The session's info string is set to "Created by Portlet Wrapper".

In Application Designer, the server side counterpart of a page that runs inside the browser is also called "adapter object".

From the adapter you can directly access the portlet request which comes from the portal server:

- The method `PortletWrapper.findPortletRequest(this)` returns an object of type `PortletRequest`.

From the portlet request you can navigate to

- the portlet session
- the portlet credential information
- the portlet context

following the Portlet 1.0 specification.

Portlet Integration and AJAX

AJAX technology is used inside Application Designer to ensure that pages in the browser are not permanently fully reloaded when a page communicates to its server side application.

Portal pages take a different approach: portal pages are always fully reloaded (fully means: all portlets that are rendered inside one page are reloaded) whenever one portlet communicates to its server side application.

The portlet integration of Application Designer takes this approach into account:

- Whenever the Application Designer portlet exchanges information with its server side counter part, NO reloading of the whole portal page is done internally. You can imagine the portlet page to be an "island" whose communication is decoupled from the portal server's page updating.

This ensures that complex pages requiring AJAX for supporting a high frontend interactivity are also usable within a portal environment.